

```

function [assignment,cost] = munkres(costMat)
% MUNKRES    Munkres (Hungarian) Algorithm for Linear Assignment Problem.
%
% [ASSIGN,COST] = munkres(COSTMAT) returns the optimal column indices,
% ASSIGN assigned to each row and the minimum COST based on the
assignment
% problem represented by the COSTMAT, where the (i,j)th element
represents the cost to assign the jth
% job to the ith worker.
%
% Partial assignment: This code can identify a partial assignment is a
full
% assignment is not feasible. For a partial assignment, there are some
% zero elements in the returning assignment vector, which indicate
% un-assigned tasks. The cost returned only contains the cost of
partially
% assigned tasks.

% This is vectorized implementation of the algorithm. It is the fastest
% among all Matlab implementations of the algorithm.

% Examples
% Example 1: a 5 x 5 example
%{
[assignment,cost] = munkres(magic(5));
disp(assignment); % 3 2 1 5 4
disp(cost); %15
%}
% Example 2: 400 x 400 random data
%{
n=400;
A=rand(n);
tic
[a,b]=munkres(A);
toc           % about 2 seconds
%}
% Example 3: rectangular assignment with inf costs
%{
A=rand(10,7);
A(A>0.7)=Inf;
[a,b]=munkres(A);
%}
% Example 4: an example of partial assignment
%{
A = [1 3 Inf; Inf Inf 5; Inf Inf 0.5];
[a,b]=munkres(A)
%}
% a = [1 0 3]
% b = 1.5
% Reference:
% "Munkres' Assignment Algorithm, Modified for Rectangular Matrices",
% http://csclab.murraystate.edu/bob.pilgrim/445/munkres.html

% version 2.3 by Yi Cao at Cranfield University on 11th September 2011

assignment = zeros(1,size(costMat,1));
cost = 0;

```

```

validMat = costMat == costMat & costMat < Inf;
bigM = 10^(ceil(log10(sum(costMat(validMat))))+1);
costMat(~validMat) = bigM;

% costMat(costMat~=costMat)=Inf;
% validMat = costMat<Inf;
validCol = any(validMat,1);
validRow = any(validMat,2);

nRows = sum(validRow);
nCols = sum(validCol);
n = max(nRows,nCols);
if ~n
    return
end

maxv=10*max(costMat(validMat));

dMat = zeros(n) + maxv;
dMat(1:nRows,1:nCols) = costMat(validRow,validCol);

%*****
% Munkres' Assignment Algorithm starts here
%*****

%%%%%%%%%%
% STEP 1: Subtract the row minimum from each row.
%%%%%%%%%%
minR = min(dMat,[],2);
minC = min(bsxfun(@minus, dMat, minR));

%*****
**
% STEP 2: Find a zero of dMat. If there are no starred zeros in its
% column or row start the zero. Repeat for each zero
%*****
**
zP = dMat == bsxfun(@plus, minC, minR);

starZ = zeros(n,1);
while any(zP(:))
    [r,c]=find(zP,1);
    starZ(r)=c;
    zP(r,:)=false;
    zP(:,c)=false;
end

while 1
%*****
**
% STEP 3: Cover each column with a starred zero. If all the columns are
% covered then the matching is maximum
%*****
**
    if all(starZ>0)
        break
    end
    coverColumn = false(1,n);

```

```

        coverColumn(starZ(starZ>0))=true;
        coverRow = false(n,1);
        primeZ = zeros(n,1);
        [rIdx, cIdx] =
find(dMat(~coverRow,~coverColumn)==bsxfun(@plus,minR(~coverRow),minC(~coverColumn)));
        while 1

%*****
**
        % STEP 4: Find a noncovered zero and prime it. If there is no
starred
        % zero in the row containing this primed zero, Go to
Step 5.
        % Otherwise, cover this row and uncover the column
containing
        % the starred zero. Continue in this manner until there
are no
        % uncovered zeros left. Save the smallest uncovered
value and
        % Go to Step 6.

%*****
**

        cR = find(~coverRow);
        cC = find(~coverColumn);
        rIdx = cR(rIdx);
        cIdx = cC(cIdx);
        Step = 6;
        while ~isempty(cIdx)
            uZr = rIdx(1);
            uZc = cIdx(1);
            primeZ(uZr) = uZc;
            stz = starZ(uZr);
            if ~stz
                Step = 5;
                break;
            end
            coverRow(uZr) = true;
            coverColumn(stz) = false;
            z = rIdx==uZr;
            rIdx(z) = [];
            cIdx(z) = [];
            cR = find(~coverRow);
            z = dMat(~coverRow,stz) == minR(~coverRow) + minC(stz);
            rIdx = [rIdx(:);cR(z)];
            cIdx = [cIdx(:);stz(ones(sum(z),1))];
        end
        if Step == 6
            %
%*****
% STEP 6: Add the minimum uncovered value to every element of
each covered
            % row, and subtract it from every element of each
uncovered column.
            % Return to Step 4 without altering any stars,
primes, or covered lines.

```

```

%*****
**

[minval,rIdx,cIdx]=outerplus(dMat(~coverRow,~coverColumn),minR(~coverRow)
,minC(~coverColumn));
    minC(~coverColumn) = minC(~coverColumn) + minval;
    minR(coverRow) = minR(coverRow) - minval;
    else
        break
    end
end
end

%*****
**

    % STEP 5:
    % Construct a series of alternating primed and starred zeros as
    % follows:
    % Let Z0 represent the uncovered primed zero found in Step 4.
    % Let Z1 denote the starred zero in the column of Z0 (if any).
    % Let Z2 denote the primed zero in the row of Z1 (there will always
    % be one). Continue until the series terminates at a primed zero
    % that has no starred zero in its column. Unstar each starred
    % zero of the series, star each primed zero of the series, erase
    % all primes and uncover every line in the matrix. Return to Step
3.

%*****
**

    rowZ1 = find(starZ==uZc);
    starZ(uZr)=uZc;
    while rowZ1>0
        starZ(rowZ1)=0;
        uZc = primeZ(rowZ1);
        uZr = rowZ1;
        rowZ1 = find(starZ==uZc);
        starZ(uZr)=uZc;
    end
end

% Cost of assignment
rowIdx = find(validRow);
colIdx = find(validCol);
starZ = starZ(1:nRows);
vIdx = starZ <= nCols;
assignment(rowIdx(vIdx)) = colIdx(starZ(vIdx));
pass = assignment(assignment>0);
pass(~diag(validMat(assignment>0,pass))) = 0;
assignment(assignment>0) = pass;
cost = trace(costMat(assignment>0,assignment(assignment>0)));

function [minval,rIdx,cIdx]=outerplus(M,x,y)
ny=size(M,2);
minval=inf;
for c=1:ny
    M(:,c)=M(:,c)-(x+y(c));
    minval = min(minval,min(M(:,c)));
end

```

```
[rIdx,cIdx]=find(M==minval);
```