HOMEWORK
*David F. Gleich*
*September 2, 2025*

# Homework 1

Please answer the following questions in complete sentences in a clearly prepared manuscript and submit the solution by the due date on Gradescope.

Remember that this is a graduate class. There may be elements of the problem statements that require you to fill in appropriate assumptions. You are also responsible for determining what evidence to include. An answer alone is rarely sufficient, but neither is an overly verbose description required. Use your judgement to focus your discussion on the most interesting pieces. The answer to "should I include 'something' in my solution?" will almost always be: Yes, if you think it helps support your answer.

## Problem 0: Homework checklist

- Please identify anyone, whether or not they are in the class, with whom you discussed your homework. This problem is worth 1 point, but on a multiplicative scale.
- Remember to disclose all use of AI tools.
- Make sure you have included your source-code and prepared your solution according to the most recent Campuswire note on homework submissions.

## Problem 1: Operations

1. Diagonal scaling. $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 6 & 4 & 2 \\ -2 & -2 & -2 \end{bmatrix} = ?$
2. Diagonal scaling. $\begin{bmatrix} 1 & 2 & 3 \\ 6 & 4 & 2 \\ -2 & -2 & -2 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & -1 \end{bmatrix} = ?$
3. Diagonal scaling. $\begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 6 & 4 & 2 \\ -2 & -2 & -2 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & -1 \end{bmatrix} = ?$
4. Describe the difference in multiplying a diagonal matrix on the left compared with the right.
5. We often use the vector $\mathbf{e} = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T$ (i.e. the all ones vector). Suppose \$$\mathbf{x}$= $\mathbf{e}$= \$ `ones(1000,1)` $\mathbf{y}$ = `1:1000`, what is $\mathbf{x}^T\mathbf{y} = ?$
6. Let $\mathbf{e}_i$ be the vector with all zeros and a 1 in the $i$th entry. Let $\boldsymbol{A}$ be an $m \times n$ matrix. Give an expression for the $r$th row of $\boldsymbol{A}$ as a result of a matrix vector product. Give an expression for the $c$th column of $\boldsymbol{A}$ as a result of a matrix vector product.
7. $\mathbf{x} = \begin{bmatrix} -5 & 4 & 2 \end{bmatrix}^T$. (Assume $\mathbf{e}_i$ is $3 \times 1$.) $\mathbf{e}_2\mathbf{x}^T =?$
   $\mathbf{x}\mathbf{e}_1^T =?$
8. Let $\boldsymbol{A} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & -1 \end{bmatrix}$. What are $\boldsymbol{A}\mathbf{e}_1$, $\boldsymbol{A}\mathbf{e}_2$, $\boldsymbol{A}\mathbf{e}_3$.

## Problem 2: Identify matrix structure

1. Consider a matrix from the social security database consisting of the GPT-5 embedding of everyone's social security number. Each row is a set of 3072 dimensions corresponding to the embedding of the string of their social security number. Explain why this is or isn't a matrix, using the ideas from class.

2. An audio file is just a big long vector of information. (Well, two vectors for a stereo audio file if you want to be very precise, but let's suppose we just have one signal.) Suppose we create a matrix where each column represents 44100 sequential pieces of information from the big vector, i.e.,

$$\boldsymbol{A} = \begin{bmatrix} x_1 & x_{44101} & x_{88201} & \cdots \\ \vdots & \vdots & \vdots & \cdots \\ x_{44100} & x_{88200} & x_{132300} & \cdots \end{bmatrix}.$$

3. Consider a matrix of demographic information for many all undergrad CS applicants at Purdue. Each student is a row. The columns represent:

- 0/1 (had a highschool GPA >= 3.75)
- 0/1 (had a highschool GPA >= 3.25)
- The numeric ID the state they graduated high-school in (there is a canonical labeling of the 50 states from 01 to 50).
- 0/1 has more than three letters of recommendation
- number of times the word "excellent" "top-tier" "best" appears in letters of recommendation
- Flesch-Kincaid Grade Level of personal statement *This example is entirely ficticious.* Explain why this is or isn't a matrix, using the ideas from class.

4. Let $\boldsymbol{A}$ be the matrix from part 1. Consider the new matrix $\boldsymbol{B} = \boldsymbol{A}^T\boldsymbol{A}$. Explain why this is or isn't a matrix, using the ideas from class.

5. Let $\boldsymbol{A}$ be the matrix from part 2. Consider the new matrix $\boldsymbol{B} = \boldsymbol{A}^T\boldsymbol{A}$. Explain why this is or isn't a matrix, using the ideas from class.

6. Let $\boldsymbol{A}$ be the matrix from part 3. Consider the new matrix $\boldsymbol{B} = \boldsymbol{A}^T\boldsymbol{A}$. Explain why this is or isn't a matrix, using the ideas from class.

## Problem 3. Some simple structure.

1. Show that the product of two diagonal matrices is also diagonal.

2. Show that the product two circulant matrices is circulant.

3. Show that multiplying a tridiagonal matrix (i.e. a matrix with entries on the diagonal and the sub and super diagonal) by itself results in a matrix with five diagonals.

## Problem 4: More interesting proofs

Consider the following problem, which I ran into when working on erasure coded eigensolvers (don't worry about what that is), let $\boldsymbol{C}$ be an $n \times k$ matrix with $n \geq k$.

1. Show that $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{C}^T \end{bmatrix}$ is invertible for any $\boldsymbol{C}$.
2. Let $\mathbf{b} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}$, give an algorithm to solve $\boldsymbol{A}\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \mathbf{b}$.
3. Give an explicit form for the inverse (this may involve the inverse of other matrices.) (Hint, show this for $\boldsymbol{C} = 0$, then try $\boldsymbol{C}$ as a rank-1 matrix $\boldsymbol{C} = \mathbf{u}\mathbf{v}^T$.)

4. *Elementary matrices* Householder, who we will talk about in forthcoming lectures, has a few things named after him. He discussed the idea that any matrix:
$$\boldsymbol{I} - \sigma \mathbf{u}\mathbf{v}^T$$
should be called an elementary matrix. Show when an elementary matrix is invertible and give the inverse.

This is what we mean by using structure to solve a problem better. We can build a simple algorithm to solve this type of system assuming we know its structure.

## Problem 5: Viral spreading in Julia

**This problem will be more difficult if you haven't used Julia or Matlab before, so get started early! It's designed to teach you about writing `for` loops to construct a matrix operation for a particular task.**

In this problem, we will explore some simple viral spreading processes in julia, both as nonlinear probability processes as well as matrix computations.

We are going to use a standard routine to generate random graphs for these experiments.

```
## Generate a simple spatial graph model to look at.
using NearestNeighbors, Distributions, SparseArrays
function spatial_graph_edges(n::Integer,d::Integer;degreedist=LogNormal(log(4),1))
  xy = rand(d,n)
  T = BallTree(xy)
  # form the edges for sparse
  ei = Int[]
  ej = Int[]
  for i=1:n
    deg = min(ceil(Int,rand(degreedist)),n-1)
    idxs, dists = knn(T, xy[:,i], deg+1)
    for j in idxs
      if i != j
        push!(ei,i)
        push!(ej,j)
      end
    end
  end
  return xy, ei, ej
end
function spatial_network(n::Integer, d::Integer; degreedist=LogNormal(log(3),1))
  xy, ei, ej = spatial_graph_edges(n, d;degreedist=degreedist)
  A = sparse(ei,ej,1,n,n)
  return max.(A,A'), xy
end
using Random
Random.seed!(10) # ensure repeatable results...
A,xy = spatial_network(10, 2)
```

1. Run the code above and report the adjacency matrix that it generates along with the coordinates.

2. Now, I like pictures! Here is the code to draw the graph given the coordinates of each node xy.

```
using Plots
function plotgraph(A::SparseMatrixCSC,xy::AbstractArray{T,2};kwargs...) where T
  px,py = zeros(T,0),zeros(T,0)
  P = [px,py]
  rows = rowvals(A)
  skip = NaN.*xy[:,begin] # first row
  for j=1:size(A,2) # for each column
    for nzi in nzrange(A, j)
      i = rows[nzi]
      if i > j
        push!.(P, @view xy[:,i])
        push!.(P, @view xy[:,j])
        push!.(P, skip)
      end
    end
  end
  plot(px,py;framestyle=:none,legend=false,kwargs...)
end
```

Use the following code to prepare a visual.

```
plotgraph(A,xy,alpha=0.25)
scatter!(xy[1,:],xy[2,:],
  markersize=2, markerstrokewidth=0, color=1)
```

Save this as a figure using savefig and include it in your writeup.

(Optional) If you want to add labels, here's a fancy way to do it.

```
annotate!.(map( i->(xy[1,i], xy[2,i], text("$i", :white, 8)), 1:size(xy,2)))
```

This is fancy syntax, so let me explain it. `annotate!` The `!` identifies a 'mutating' function because it changes something. The `.` means "broadcast" over the list. That is equivalent to running `annotate!` for each element of the list. The `map` function just builds the list to run `annotate!` over by telling it the x,y coordinate of where to annotate and then what to annotate (in this case, text, which gives the value `i` that is input to the map function, in white, at font-size 8. You may need to change 8 depending on your display.

3. Next up, we want to run our evolving viral spreading process. Here, we are going to evolution process that does not consider the probability of infecting yourself based on the previous time-step. We are also going to remove the max and handle it in a slightly different way. Fill in the missing code to do this.

```
function evolve(x::Vector,  p::Real, A::AbstractMatrix)
  log_not_infected = log.(1 .- p.*x)
  y = 1 .- exp.(A*log_not_infected)
end
"""
Run k steps of the evolution and return the results as a matrix.
Each column of the matrix has the probabilities that the node
is infected under the `wrong` evolve function.
The first column of X is the initial vector x0.
At each iteration, we make sure the probabilities are at least x0 and these
are fixed.
"""
function evolve_steps(x0::Vector, p::Real, A::AbstractMatrix, k::Int)
  X = zeros(length(x0),k+1)
  # fill in the missing code
    X[:,i+1] = max.(evolve(X[:,i], p, A), X[:,1]) # fix the initial probability x0
  # fill in the missing code
  return X
end
```

Show the matrix X and explain our starting condition where we set:

```
p = 0.2
x0 = zeros(size(A,1))
x0[1] = 1
evolve_steps(x0, p, A, 10)
```

4. Given such a matrix $X$, the column-sums reflect the expected number of people that are infected (this is true if we make the slightly unrealistic assumption that people combine linearly in expectation and are independent – both bad, but simple, assumptions.) (i) Make a plot of the column sums of X for all rows of X except for row 1, which is the node that we definitely infected in the previous part. (Hint: this is a partial answer to the previous question, good job reading ahead!) (ii) Do you agree or disagree that this plot intuitively shows the expected number of infections possible from this single infection in expectation in our very simple model? (If you like to be extremely detailed about probability calculations–and it is good to be that way, but I can't put all possible details in one simple question unfortunately–put on a physisicst hat for a second and pretend to be a little more forgiving.) Explain your reasoning.

5. First, (i) implement the following function for our approximation evolution.

```
"""
Run k steps of the approximate evolution and return the results as a matrix.
Each column of the matrix has the probabilities that the node
is infected under the `wrong` evolve function.
The first column of X is the initial vector x0.
At each iteration, we make sure the probabilities are at least x0 and these
are fixed.
"""
function approx_evolve_steps(x0::Vector, p::Real, A::AbstractMatrix, k::Int)
  X = zeros(length(x0),k+1)
  # fill in the code using the approximation evolution.
  return X
end
```

and (ii) also make the same plot as in part 4 and compare to your previous plot. Finally, (iii) find the largest value of p or $\rho$ such that the two plots look similar to you!

Here was some of my code to play around with this.

```
p = 0.2
x0 = zeros(size(A,1))
x0[1] = 1
Y = approx_evolve_steps(x0, p, A, 15)
X = evolve_steps(x0, p, A, 15)
plot(sum(X[2:end,:],dims=1)')
plot!(sum(Y[2:end,:],dims=1)')
```

6. Now, we are going to make the networks much bigger. Let's make a 1000 node network. We will also start the infections from node 1000.

```
Random.seed!(10)
A,xy = spatial_network(1000, 2)
```

Assume the column-sums represent an appropriate expected value for the number of infections at time step $k$ due to the single initial infection. (If you listed caveats above, consider them noted, but let's keep it simple here!)

How many people would we expect to be infected under the true probability model vs. the approx. model for $p = 0.05, 0.1, 0.15, 0.20$ after 10 steps where node 1000 is infected initially?

7. Some of those numbers are big! We want fewer people infected. Let's try social distancing. Suppose each person visits half of the number of people, where we pick the folks to keep based on our coordinates xy. (We keep the closest people to us, of course...)

```
""" return a new "social" graph where we have implemented
social distancing by removing an f-fraction of your neighbors
based on spatial proximity. So f=0 is the original network
and f=1 is the empty network."""
function social_distance(A::SparseMatrixCSC, xy::Matrix, f::Real)
  # access the CSC arrays directly, see help on nzrange
  rowval = rowvals(A)
  n = size(A,1)
  new_ei = Vector{Int}()
  new_ej = Vector{Int}()
  for j = 1:n
    neighbors = Vector{Int}()
    dists = Vector{Float64}()
    myxy = @view xy[:,j] # don't make a copy
    for nzi in nzrange(A, j)
      # edge from (i,j)
      i = rowval[nzi]
      push!(neighbors, i)
      push!(dists, norm(xy[:,i]-myxy))
    end
    p = sortperm(dists) # sort distances
    nkeep = ceil(Int, (1-f)*length(dists))
    for i=1:nkeep
      push!(new_ei, neighbors[p[i]])
      push!(new_ej, j)
    end
  end
  A = sparse(new_ei,new_ej,1, size(A,1),size(A,2))
  return max.(A,A')
end
```

How do various levels of social distancing change the expected number of people infected by node 1000 after 10 steps? Lets say 10% distancing, up to 60%. Evaluate this both for the "exact" vs. "approximate" model.

8. What would wearing masks change in this model? Can you work out anything analytical about how wearing masks changes the approximate model?

## Problem 6: The expected length of a Chutes and Ladders game

**This problem will be more difficult if you haven't used Julia or Matlab before, so get started early! It's designed to teach you about writing `for` loops to construct a matrix operation for a particular task.**

In class, we showed that we could form a linear system of equations in order to determine the expected time that a random walk on the integers $[-4, 6]$ spends before it reaches the endpoints $-4$ and 6. We can do the same thing to determine the expected length of a game of Chutes and Ladders!

The data for Chutes and Ladders as a Markov chain is available from

- https://www.cs.purdue.edu/homes/dgleich/cs515-2025/homeworks/chutes-and-ladders-matrix.csv
- https://www.cs.purdue.edu/homes/dgleich/cs515-2025/homeworks/chutes-and-ladders-coords.csv

The starting state is state 101. The ending state is 100.

(This is based on https://www.datagenetics.com/blog/november12011/)

1. Use the same type of implicit formulation where $x_i$ is the expected length of a Chutes and Ladders game starting from cell $i$ to derive a linear system of equations. Build and solve this linear system of equations in Julia. You should provide the linear system in terms of the matrix $\boldsymbol{T}$, but you should not provide explicit entries. Also, presumably, the creators would have made the start state the place of maximum game length. Is this the case or is there another cell or set of cells that result in a longer game?

   For this problem, I found it helpful to visualize a solution to make sure it made sense. I used the following line of code to visualize a solution $\mathbf{x}$:

```
p = scatter(xc, yc, zcolor=x,
     markersize=16, label="", marker=:square, markerstrokewidth=0, size=(400,400),
    xlims=(0.5,10.5),ylims=(0.5,10.5),aspect_ratio=:equal)
function draw_chutes_and_annotate(p)
    CL = [ 1   4   9  21  28  36  51  71   80  98  95  93  87  64  62  56  49  48  16
          38  14  31  42  84  44  67  91  100  78  75  73  24  60  19  53  11  26   6]
    for col=1:size(CL,2)
        i = CL[1,col]
        j = CL[2,col]
        if i > j # this is a chute
            plot!(p,[xc[i],xc[j]],[yc[i],yc[j]],color=2,label="")
        else
            plot!(p,[xc[i],xc[j]],[yc[i],yc[j]],color=1,label="")
        end
    end
    map(i->annotate!(p,xc[i],yc[i], text("$i", :white, 8)), 1:100)
    p
end
draw_chutes(p)
```

   where `xc` and `yc` are the coordinates from the coords file.

2. Recall that in class we showed that $\mathbf{p}_k = \boldsymbol{T}^{k-1}\mathbf{t}_{101}$ gave the probability of being in each state after $k$ steps. By definition, the expected length of the game is:

$$\sum_{k=1}^{\infty} k[\mathbf{p}_k]_{100}.$$

   Develop a computer program to approximately compute this sum. Is there a good way to determine when to stop adding terms? What value do you get?

## Problem 7: Poisson's equation

**This problem will be more difficult if you haven't used Julia or Matlab before, so get started early! It's designed to teach you about writing**

**`for` loops to construct a matrix operation for a particular task.**

**Also note that you will use this routine on future homeworks. Make sure to talk to us in Office hours if you don't get it right.**

In this problem, we'll meet one of the most common matrices studied in numerical linear algebra: the $2d$-Laplacian. We arrive at this matrix by discretizing a partial differential equation. Poisson's equation is:

$$\Delta u = f$$

where $u(x, y)$ is a continuous function defined over the unit-plane (i.e. $0 \leq x \leq 1, 0 \leq y \leq 1$), $f(x, y)$ is a continuous function defined over the same region, and $\Delta$ is the Laplacian operator:

$$\Delta u = \partial^2 u/\partial x^2 + \partial^2 u/\partial y^2.$$

Given a function $f$, we want to find a function $u$ that satifies this equation. There are many approaches to solve this problem theoeretically and numerically. We'll take a numerical approach here.

Suppose we discretize the function $u$ at regular points $x_0, \ldots, x_n$, and $y_0, \ldots, y_n$ where $x_i = y_i = i/n$ so that we have:

$$u(x,y) \approx \text{ grid of } \begin{matrix} u(x_0, y_0) & \cdots & u(x_n, y_0) \\ \vdots & \ddots & \vdots \\ u(x_0, y_n) & \cdots & u(x_n, y_n). \end{matrix}$$

For this discretization, note that

$$\begin{aligned} \Delta u(x_i, y_j) &\approx n^2 \left(u(x_{i-1}, y_j) - 2u(x_i, y_j) + u(x_{i+1}, y_j)\right) \\ &\qquad + n^2 \left(u(x_i, y_{j-1}) - 2u(x_i, y_j) + u(x_i, y_{j+1})\right) \\ &= n^2 \left(u(x_{i-1}, y_j) + u(x_i, y_{j-1}) - 4u(x_i, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j+1})\right) \\ &= f(x_i, y_j). \end{aligned}$$

What we've done here is use the approximation:

$$\partial^2 u/\partial x^2 \approx \frac{1}{h^2} \left(u(x-h) - 2u(x) + u(x+h)\right)$$

for both partial terms.
We need this equation to hold at each point $x_i, y_j$. But note that there are some issues with this equation at the boundary values (where x=0 or 1, or where y=0 or 1).
For this problem, we'll make it very simple and set:

$$u(0, y_j) = u(1, y_j) = u(x_i, 0) = u(x_i, 1) = 0.$$

Now, we'll do what we always do! Turn this into some type of matrix equation!

Let $\boldsymbol{U}$ be an $n+1 \times n+1$ matrix that we'll index from zero instead of one:

$$\boldsymbol{U} = \begin{bmatrix} U_{0,0} & \cdots & U_{0,n} \\ \vdots & \ddots & \vdots \\ U_{n,0} & \cdots & U_{n,n} \end{bmatrix}.$$

where $U_{i,j} = u(x_i, y_j)$. At this point, we are nearly done. What we are going to do is turn Poisson's equation into a linear system.

In order to write $\boldsymbol{U}$ as a vector, we'll keep the convention from last time:

$$\boldsymbol{U} = \begin{bmatrix} u_1 & \cdots & u_{n+1} \\ u_{n+2} & \cdots & u_{2(n+1)} \\ \vdots & \ddots & \vdots \\ u_{n(n+1)+1} & \cdots & u_{(n+1)(n+1)} \end{bmatrix}.$$

Let $\mathbf{u}$ be the vector of elements here. Note that our approximation to $\Delta u$, just involved a linear combination of the elements of $\mathbf{u}$. This means we have a linear system:

$$\boldsymbol{A}\mathbf{u} = \mathbf{f}$$

where the rows of $\boldsymbol{A}$ and $\mathbf{f}$ correspond to equations of the form:

$$\frac{1}{h^2}\left(u(x_{i-1}, y_j) + u(x_i, y_{j-1}) - 4u(x_i, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j+1})\right) = f(x_i, y_j).$$

1. Let $n = 3$. Write down the $16 \times 16$ linear equation for $\mathbf{u}$ including all the boundary conditions. Note that you can encode the boundary conditions by adding a row of $\boldsymbol{A}$ where: $u_i = 0$.

2. Write a Julia or Matlab/Python code to construct a sparse matrix $\boldsymbol{A}$ and vector $\mathbf{f}$ when $n = 10$ and $f(x, y) = 1$. Here's some pseudo-code to help out:

```
function laplacian(n::Integer, f::Function)
    N = (n+1)^2
    nz = <fill-in>
    I = zeros(Int,nz)
    J = zeros(Int,nz)
    V = zeros(nz)
    fvec = zeros(N)
    # the transpose mirrors the row indexing we had before.
    G = reshape(1:N, n+1, n+1)' # index map, like we saw before;
    h = 1.0/(n)
    index = 1
    for i=0:n
        for j=0:n
            row = G[i+1,j+1]
            if i==0 || j == 0 || i == n || j == n
                # we are on a boudnary
                fvec[row] = 0.0
                # fill in entries in I,J,V and update index
            else
                fvec[row] = f(i*h, j*h)*h^2
                # fill in entries in I,J,V and update index
            end
        end
    end
    A = sparse(I,J,V,N,N)
    return A, fvec
end
```

3. Solve for $\mathbf{u}$ using Julia's or Matlab's backslash solver, and show the result using the `mesh` function (Matlab) or `surface` function (Plots.jl in Julia).

4. Show that we can eliminate rows and columns from the matrix and or equations and variables from the problem for the nodes on the boundary. The resulting matrix should be symmetric. This is what we will call the "poisson" matrix in future problems in class.

## Problem 8: Sparse matrix operations

Write working code for the following operations for a matrix given by Compressed Sparse Column arrays: `colptr`, `rowval`, `nzval` along with dimensions `m` and `n` as in the Julia sparse matrix storage.

1. Sparse matrix-transpose multiplication by a vector

```
""" Returns y = A'*x where A is given by the CSC arrays
colptr, rowval, nzval, m, n and x is the vector. """
function csc_transpose_matvec(colptr, rowval, nzval, m, n, x)
end
```

2. Column inner-product

```
""" Returns  = A[:,i]'*x where A is given by the CSC arrays
colptr, rowval, nzval, m, n and x is the vector. """
function csc_column_projection(colptr, rowval, nzval, m, n, i, x)
end
```

3. Column-column inner-product

```
""" Returns rho = A[:,i]'*A[:,j] where A is given by the CSC arrays
colptr, rowval, nzval, m, n and i, and j are the column indices. """
function csc_col_col_prod(colptr, rowval, nzval, m, n, i, j)
end
```

4. Lookup element

```
""" Returns rho = A[i,j] where A is given by the CSC arrays
colptr, rowval, nzval, m, n and i, and j are the column indices. """
function csc_lookup(colptr, rowval, nzval, m, n, i, j)
end
```

5. Lookup row

```
""" Returns x = A[i,:] where A is given by the CSC arrays
colptr, rowval, nzval, m, n and i is the row index . """
function csc_lookup_row(colptr, rowval, nzval, m, n, i)
end
```