1)

Given that:

**U** is the orthogonal m x m matrix with columns

**V** is the orthogonal n x n matrix with columns with transpose **V**T

**Σ** is the diagonal m x n matrix with values

We can then say by definition:

This can then be extrapolated into:

Because U is orthogonal, we can then say:

Taking the following:

We derive the general form

Taking i = 1, we observe that:

, q.e.d.,

2)

**Calculations:**

Given the directed graph in the problem, we can construct the adjacency matrix of outward edges, A, as such with the following ordering:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AI | Stats | DSci | DMin | DEng | DVis | Prog | PComp | ML | Julia |  |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | AI |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | Stats |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | DSci |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | DMin |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | Deng |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | DVis |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | Prog |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | PComp |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ML |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | Julia |

We can then construct the diagonal matrix D by summing the values in each row to get the following:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AI | Stats | DSci | DMin | DEng | DVis | Prog | PComp | ML | Julia |  |
| 3 |  |  |  |  |  |  |  |  |  | AI |
|  | 3 |  |  |  |  |  |  |  |  | Stats |
|  |  | 6 |  |  |  |  |  |  |  | DSci |
|  |  |  | 1 |  |  |  |  |  |  | DMin |
|  |  |  |  | 1 |  |  |  |  |  | Deng |
|  |  |  |  |  | 1 |  |  |  |  | DVis |
|  |  |  |  |  |  | 2 |  |  |  | Prog |
|  |  |  |  |  |  |  | 1 |  |  | PComp |
|  |  |  |  |  |  |  |  | 1 |  | ML |
|  |  |  |  |  |  |  |  |  | 1 | Julia |

We observe that none of the values in the D matrix have a 0 value, so no shifting is required.

To solve the equation , we find the transpose of the adjacency matrix A and the inverse of the diagonal matrix D. We utilize Julia to perform the numerical calculations as follows, taking values of alpha from 0 to 0.9999 (due to the limitations of the matrix network package) in steps of 0.0001, and observing the unique rankings that occur.

**Code and Results:**

using MatrixNetworks

using SparseArrays

using LinearAlgebra

## define iterable list and ordering list

a\_iter = 0:0.0001:0.9999

orderings\_man = Set{Vector{Int}}()

orderings\_pkg = Set{Vector{Int}}()

## Define the adjacency matrix A

A = [0 1 1 0 0 0 0 0 1 0; ## AI

1 0 1 1 0 0 0 0 0 0; ## Statistics

1 1 0 1 0 1 1 0 0 1; ## Data Science

0 0 0 0 1 0 0 0 0 0; ## Data Mining

0 0 0 0 0 1 0 0 0 0; ## Data Engineering

0 0 0 0 0 0 1 0 0 0; ## Data Visualization

0 0 0 0 0 0 0 1 0 1; ## Programming

0 0 0 0 0 0 1 0 0 0; ## Parallel Computing

1 0 0 0 0 0 0 0 0 0; ## Machine Learning

0 0 0 0 0 0 0 0 1 0]## Julia

sparse\_A = sparse(A)

## define the pageRank function

function pageRank(adj\_mat::Matrix, alpha::Float64)

n = size(adj\_mat, 1)

## Create diagonal matrix

d\_mat = Diagonal(sum(adj\_mat, dims=1)[:])

d\_inv = Diagonal(1 ./ diag(d\_mat))

## Create I - alpha\*D^-1\*A'

lhs\_mat = I - alpha \* adj\_mat' \* d\_inv

## Define the right-hand side vector

rhs\_vec = (1- alpha) / n \* ones(n)

## Solve the equation

x = lhs\_mat \ rhs\_vec

return x

end

## Calculate pageRank manually and through MatrixNetworks

for a in a\_iter

p = sortperm(pageRank(A, a), rev=true)

push!(orderings\_man, p)

p = sortperm(pagerank(sparse\_A, a), rev=true)

push!(orderings\_pkg, p)

end

# Print results

for ordering in orderings\_pkg

@printf("Ordering: %s\n", join(ordering, ", "))

end

@printf("MatrixNetworks Ranking Count: %.0f\n\n", length(orderings\_pkg))

for ordering in orderings\_man

@printf("Ordering: %s\n", join(ordering, ", "))

end

@printf("Manual Calculation Ranking Count: %.0f\n", length(orderings\_man))

|  |  |
| --- | --- |
| Ordering: 1, 7, 9, 10, 3, 8, 6, 2, 5, 4  Ordering: 7, 1, 9, 10, 6, 8, 3, 5, 2, 4  Ordering: 1, 7, 9, 10, 3, 2, 8, 6, 5, 4  Ordering: 7, 1, 9, 6, 10, 5, 3, 8, 2, 4  Ordering: 1, 7, 9, 10, 8, 3, 6, 2, 5, 4  Ordering: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10  Ordering: 7, 1, 9, 6, 10, 5, 8, 3, 2, 4  Ordering: 7, 1, 9, 10, 8, 6, 3, 2, 5, 4  Ordering: 1, 9, 7, 3, 10, 2, 8, 6, 5, 4  Ordering: 7, 1, 9, 6, 5, 10, 3, 8, 2, 4  Ordering: 7, 1, 9, 6, 10, 8, 3, 5, 2, 4  Ordering: 7, 1, 9, 6, 10, 8, 5, 3, 2, 4  Ordering: 7, 1, 9, 10, 6, 8, 3, 2, 5, 4  Ordering: 7, 1, 9, 10, 8, 3, 6, 2, 5, 4  Ordering: 1, 9, 7, 10, 3, 2, 8, 6, 5, 4  Ordering: 1, 7, 9, 10, 3, 8, 2, 6, 5, 4  MatrixNetworks Ranking Count: 16 | Ordering: 7, 1, 6, 10, 4, 9, 3, 2, 8, 5  Ordering: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10  Ordering: 7, 1, 6, 10, 4, 9, 2, 3, 8, 5  Ordering: 7, 1, 6, 10, 9, 4, 3, 2, 8, 5  Ordering: 7, 1, 6, 4, 10, 9, 3, 2, 5, 8  Ordering: 7, 1, 6, 10, 4, 9, 3, 2, 5, 8  Ordering: 7, 1, 6, 10, 9, 4, 2, 3, 8, 5  Ordering: 7, 1, 6, 4, 10, 9, 2, 3, 5, 8  Ordering: 7, 1, 6, 10, 4, 9, 2, 3, 5, 8  Manual Calculation Ranking Count: 9 |

**Conclusion:**

Both a manual calculation using the LinearAlgebra package and the MatrixNetworks package pagerank function were used. From the observations above, we see that there were 16 identifiable rankings using the MatrixNetworks package, and only 9 using the manual calculation of PageRank. It is interesting to note that the manual calculations of page rank do not identify any orderings that begin with node 1 other than the sequential order ranking (1, 2, …, 10).

3A)

Code and Results

Part 1:

using LinearAlgebra

using Printf

function count\_darts(N::Int, k::Int)

num\_inside = 0

for i=1:N

p = rand(k)

num\_inside += sum(abs.(p)) <= 1

end

return num\_inside

end

##

N = 1000000

for k=2:20

n = count\_darts(N, k)

println(k, " -> ", n, " | ", (n/N\*100), "%")

end

##

Results:

A screenshot of a computer

Description automatically generated

3B)

Based on the results from above, we can see that the volume of the rhomboidal space rapidly decreases with each additional dimension added to the space.

Per expectations, for the 2-dimensional space, we expect that the rhombus, as stated, takes up about 50% of the volume, which we see as 499,395 darts within the rhomboidal space. For the 3-dimensional space, this declines rapidly to only 167009 darts within the space, and so on and so forth for each additional dimension, converging to 0 at the 10th dimension.

Considering the volume of the hypercube at each dimension, , we can then observe that the rhomboidal region takes up , which is validated by the volumes observed above.