MATH 104 Data and Decisions Spring 2017

Study Sheet for Exam #2
Networks

- This study sheet will not be allowed during the exam.
- Books, notes and online resources will not be allowed during the exam.
- A calculator will be handed out for use during the exam. **No other electronic devices (calculators, cell phones, tablets, laptops, etc.) will be allowed during the exam.**

Topics

1. Euler paths and Euler circuits
2. Vertex coloring, chromatic number
3. Minimum spanning tree, Prim's Algorithm
4. Breadth-First Search Algorithm
5. Dijkstra's Algorithm
6. PageRank without matrices
7. Matrices
8. PageRank with matrices

Practice Problems from Homework Exericses

- Exercise 2.1
- Exercise 2.2
- Exercise 2.3
- Exercise 2.4
- Exercise 2.5
- Exercise 2.6
- Exercise 2.7
- Exercise 2.8
- Exercise 2.9
- Exercise 2.10
- Exercise 2.11
- Exercise 2.12
- Exercise 2.13
- Exercise 2.14
- Exercise 2.15
- Exercise 2.16
- Exercise 2.17
- Exercise 2.18
- Exercise 2.19
- Exercise 2.23
- Exercise 2.24
- Exercise 2.25
- Exercise 2.28
- Exercise 2.26

Practice Problems from In-Class Exericses

- Exercise 2.1
- Exercise 2.2
- Exercise 2.3
- Exercise 2.4
- Exercise 2.5
- Exercise 2.6
- Exercise 2.7
- Exercise 2.8
- Exercise 2.9
- Exercise 2.10
- Exercise 2.11
- Exercise 2.12
- Exercise 2.13
- Exercise 2.14
- Exercise 2.15
- Exercise 2.19
- Exercise 2.20
- Exercise 2.21
- Exercise 2.24
- Exercise 2.22
- Exercise 2.23

Tips for Studying for the Exams

×	**Bad**	Forgetting about the homework.
✓	**Good**	Making sure you know how to do all the relevant problems from the homework exercises and in-class exercises; seeking help from the instructor and the tutors about the problems you do not know how to do.
×	**Bad**	Studying only by reading the texts.
✓	**Good**	Doing exercises, and reading the texts as needed.
×	**Bad**	Studying only by yourself.
✓	**Good**	Trying some exercises by yourself (or with friends), and then seeking help from the instructor about the exercises you do not know how to do.
×	**Bad**	Doing exercises while looking everything up in the texts or summaries.
✓	**Good**	Doing some of the exercises the way you would do them on the exams, which is with closed book.
×	**Bad**	Staying up late (or all night) the night before the exam.
✓	**Good**	Studying hard up through the day before the exam, but getting a good night's sleep the night before the exam.

Ethan's Office Hours

- **Monday:** 11:00-12:30
- **Tuesday:** 2:00-3:30 & 5:00-6:00
- **Thursday:** 2:30-4:00
- **Or by appointment**
- **Extra: Friday, 7 April 2017, 1:00-4:00**

Concepts and Formulas You Need to Know for the Exam

1. Graphs

1. A **graph** G is a collection of objects, called **vertices** (singular: vertex), together with another collection of objects, called **edges**, such that each edge connects two vertices (which are not necessarily different).
2. A **loop** in a graph is an edge of the graph that has both endpoint the same.
3. The collection vertices of the graph G is denoted $V(G)$.
4. The collection edges of the graph G is denoted $E(G)$.

2. Graph Definitions

Let G be a graph.

1. Let v and w be vertices of G. The vertices v and w are **adjacent** is there is an edge containing them.
2. Let v be a vertex of G. The **degree** of the vertex v, denoted $\deg(v)$, is the number of edges that contain v. If a loop contains v, it is counted twice in the degree.
3. A graph H is a **subgraph** of G if all the vertices of H are vertices of G and all the edges of H are edges of G.

3. Components

Let G be a graph.

1. The graph G is **connected** if for every two vertices x and y, there is a path in G that starts at x and ends at y.
2. A **component** of the graph G is a subgraph of G that is connected, and is that is not contained in a larger subgraph of G that is connected.

4. Other Types of Graphs

1. A **directed graph** G is a collection of objects, called **vertices** (singular: vertex), together with another collection of objects, called **directed edges**, such that each directed edge starts at a vertex and ends at a vertex (which is not necessarily different from the starting vertex).
2. A **weighted graph** G is a graph (undirected) such that every edge is assigned a number, called the **weight** of the edge.

5. Total Weight

Let G be a weighted graph. Let H be a subgraph of G. The **total weight** of H is the sum of the weights of the edges of H.

6. Paths and Circuits

Let G be a graph.

1. A **path** in the graph G is an alternating sequence

$$v_0, e_0, v_1, e_1, v_2, \ldots, v_{k-1}, e_{k-1}, v_k$$

of vertices and edges of G, which begins and ends with vertices, such that the two vertices of each edge of the sequence are the vertices in the sequence before and after the edge.

2. A **circuit** in the graph G is a path that starts and ends at the same vertex.

7. Euler Paths and Euler Circuits

Let G be a graph.

1. An **Euler path** in the graph G is a path in the graph that includes each edge of the graph exactly once.

2. An **Euler circuit** in the graph G is a circuit in the graph that includes each edge of the graph exactly once.

8. Euler Paths and Euler Circuits Theorem

Let G be a graph.

1. The graph G has an Euler circuit if and only if every vertex of G has even degree; an Euler circuit can start and end at any vertex of G.

2. The graph G has an Euler path that is not a circuit if and only if two vertices of G have odd degree and all other vertices have even degree; an Euler past must start at one of the vertices of odd degree and end at the other vertex of odd degree.

9. Vertex Coloring

Let G be a graph. A **vertex coloring** (also called a **coloring**) of the graph G is an assignment of colors (often written as numbers) to the vertices of G such that adjacent vertices are assigned different colors.

10. Chromatic Number

Let G be a graph.

1. Let k be a positive integer. A **k-coloring** of the graph G is a coloring of G with exactly k colors.
2. The **chromatic number** of the graph G, denoted $\chi(G)$, is the smallest whole number k such that G has a k-coloring.

11. Cycle and Tree

Let G be a graph.

1. A **cycle** in the graph G is a circuit with no repeated vertices other than the starting and ending vertex.
2. The graph G is a **tree** if it is connected and it does not have any cycles.

12. Spanning Trees

Let G be a graph. A **spanning tree** of the graph G is subgraph that is a tree and that contains all the vertices of G.

13. Spanning Tree Theorem

Let G be a graph. If G is connected, then it has a spanning tree.

14. Minimum Spanning Trees

Let G be a weighted graph. Suppose that G is connected. A **minimum spanning tree** of the graph G is a spanning tree of G that has the smallest total weight of any spanning tree of G.

15. Prim's Algorithm

Let G be a weighted graph. Suppose that G is connected. This algorithm, called **Prim's Algorithm**, finds a minimum spanning tree of the graph.

1. Choose an edge of G that has the smallest weight among all the edges of G; if there is more than one such edge, choose one. Mark the chosen edge and its vertices.
2. Choose an edge of G that has the smallest weight among all the unmarked edges of G that have one vertex marked and one vertex unmarked; if there is more than one such edge, choose one. Mark the chosen edge and its vertices.
3. Repeat the above procedure until all the vertices are marked.

16. Average Degree

Let G be a graph. Suppose that G has N vertices. Let $v_1, \ldots, v_N$ be the vertices of G, and let $k_1, \ldots, k_N$ denote the degrees of the vertices, respectively. The **average degree** of G, denoted $\bar{k}$ (and also $\langle k \rangle$), is defined by

$$\bar{k} = \frac{1}{N} \sum_{i=1}^{N} k_i.$$

17. Shortest Path

Let G be a graph, and let v and w be vertices of G. A **shortest path** from v to w is a path from v to w that has the fewest edges among all paths from v to w.

18. Edge Distance

Let G be a graph, and let v and w be vertices of G. The **edge distance** (also called **distance**) from v to w is the number of edges of a shortest path from v to w.

19. Breadth-First Search Algorithm

Let G be a graph. Suppose that G is connected. Let A be a vertex of G. This algorithm, called **Breadth-First Search Algorithm** (abbreviated **BFS**), finds the edge distance from A to every other vertex of G.

The algorithm starts with the vertices of the graph unlabeled. The label on each vertex at the end of the algorithm is the edge distance from A to that vertex.

1. Label A with value 0.
2. For each vertex that is adjacent to A, label it with value 1.
3. For each vertex that is unlabeled and is adjacent to a vertex labeled 1, label it with value 2.
4. For each vertex that is unlabeled and is adjacent to a vertex labeled 2, label it with value 3.
5. Continue the above process until all vertices are labeled.
6. The numerical label of each vertex is the edge distance from A to that vertex.

20. Average Distance and Diameter

Let G be a graph.

1. The **average distance** of G is the average of the edge distances between all pairs of distinct vertices of G.
2. The **diameter** of G is the longest edge distance between any pair of distinct vertices of G.

21. Shortest Weighted Path

Let G be a weighted graph, and let v and w be vertices of G. A **shortest weighted path** from v to w is a path from v to w that has the smallest total weight among all paths from v to w.

22. Weighted Distance

Let G be a graph, and let v and w be vertices of G. The **weighted distance** (also called **distance**) between v and w is the total weight of edges of a shortest weighted path from v to w.

23. Dijkstra's Algorithm

Let G be a weighted graph. Suppose that all the edge weights are positive or zero. Suppose that G is connected. Let A be a vertex of G. This algorithm, called **Dijkstra's Algorithm**, finds the weighted distance to every other vertex of G, and finds a shortest path from A to every other vertex.

At every step of the algorithm, each vertex will be labeled with a numerical value from 0 to ∞, including possibly either of those. After each step of the algorithm, the number labeling some of the vertices might be updated to a smaller number than the previous label. The label on each vertex at the end of the algorithm is the weighted distance from A to that vertex.

1. Label A with value 0, and label every other vertex with value ∞.

2. Circle vertex A. This vertex is the currently visited vertex.

3. For each vertex that is adjacent to A, update its numerical label to the distance to A, and draw a small arrow at the vertex pointing to A.

4. Among all the vertices on G other than A, choose the one with the smallest current numerical label. Circle that vertex, and designate it as the currently visited vertex.

5. Main Step: Suppose that the currently visited vertex is labeled X. Look at all the un-circled vertices of G that are adjacent to X, one at a time (in any order). Let U be such a vertex. Add the distance from X to U to the numerical label of X. If this sum is less than the numerical label of U, then update the numerical label of U to the sum, draw a small arrow at U pointing to X, and erase the previous small arrow at U pointing to another vertex if there is such an arrow; if the sum is not less than the numerical label of U, then do not change anything for U. Do that for all un-circled vertices that are adjacent to X.

6. Among all the un-circled vertices of G, choose the one with the smallest current numerical label. Circle that vertex, and designate it as the currently visited vertex.

7. Repeat the Main Step until all the vertices are circled.

8. When every vertex is circled, each vertex will have a numerical label that is less than ∞, and each vertex other than A will have a one small arrow pointing to another vertex.

9. The final numerical label of each vertex is the weighted distance from A to that vertex.

10. To find a shortest path from A to another vertex, start at that other vertex, and follow the arrows back to A.

24. PageRank Algorithm

Let G be a directed graph. This algorithm, called the **PageRank Algorithm**, ranks the vertices of G by assigning a number to each vertex, where the vertices are then ranked from highest assigned number to lowest assigned number.

The algorithm works by making an initial assignment of numbers to the vertices, and then revising the numbers repeatedly until the desired numbers are found.

Let p be a number between 0 and 1. The number p is typically 0.85.

1. Suppose that G has n vertices. Assign each vertex an initial value of $\frac{1}{n}$.

2. If a vertex has m edges going out of it (that is, the out-degree of the vertex is m), then give every edge going out of that vertex weight $\frac{1}{m}$.

3. Redistribute the number assigned to each vertex as follows. Suppose a vertex is assigned the number x, and it has m edges going out of it. Then transfer $x \cdot \frac{1}{m}$ to each of the vertices that are reached by the m edges going out of that vertex. Do that simultaneously to all the vertices.

4. Modify the number assigned to each vertex as follows. Suppose that after the redistribution a vertex is assigned the number y. Recall that there are n vertices. Then modify the number assigned to this vertex to be $p \cdot y + (1 - p) \cdot \frac{1}{n}$. Do that simultaneously to all the vertices.

5. Repeat this two-step process (redistribution and modification, redistribution and modification, and so on) until it appears that the numbers assigned to the vertices do not change with each new repetition of the two-step process.

6. The numbers assigned to each vertex in the previous step are the final numbers assigned to each vertex.

7. A way to check for errors in calculation at each stage of the process is to use the fact that at each stage of the process, the sum of the numbers assigned to the vertices is always 1. If the sum is ever not 1, that indicates an error.

25. Matrices Definitions

1. A **matrix** (plural: **matrices**) is a rectangular array of numbers enclosed in square brackets.
2. The size of the matrix is determined by the number of rows and the number of columns.
3. If a matrix has m rows and n columns, it is called an $m \times n$ **matrix**.
4. Let A be an $m \times n$ matrix. Then A is written in general as

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

   where the entry a_{ij} is the entry in the i^{th} row and j^{th} column.
5. The entry in the i^{th} row and j^{th} column of a matrix is referred to as the (i, j) entry in the matrix.
6. A **vector** (also called a **column vector**) is an $m \times 1$ matrix.

26. Matrices: Addition and Scalar Multiplication

Let $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$ and $B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$, and let c be a number.

1. $A + B = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$.
2. $A - B = \begin{bmatrix} a_{11}-b_{11} & a_{12}-b_{12} & \cdots & a_{1n}-b_{1n} \\ a_{21}-b_{21} & a_{22}-b_{22} & \cdots & a_{2n}-b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}-b_{m1} & a_{m2}-b_{m2} & \cdots & a_{mn}-b_{mn} \end{bmatrix}$.
3. $-A = \begin{bmatrix} -a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & -a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ -a_{m1} & -a_{m2} & \cdots & -a_{mn} \end{bmatrix}$.
4. $cA = \begin{bmatrix} ca_{11} & ca_{12} & \cdots & ca_{1n} \\ ca_{21} & ca_{22} & \cdots & ca_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ ca_{m1} & ca_{m2} & \cdots & ca_{mn} \end{bmatrix}$.

27. Matrices: Multiplication

Row times Column

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = a_1 b_1 + a_2 b_2 + \cdots a_n b_n.$$

General If A is an $m \times p$ matrix and B is a $p \times n$ matrix, then AB is an $m \times n$ matrix obtained by multiplying each row in A by each column in B.

28. Adjacency Matrix of a Graph

Let G be a graph. Suppose that G has N vertices. Let $v_1, \ldots, v_N$ be the vertices of G. The **adjacency matrix** of G is the $N \times N$ matrix that has the (i, j) entry in the matrix equal to 1 if the vertices v_i and v_j are adjacent and equal to 0 if the vertices v_i and v_j are not adjacent.

29. Probability Vector and Stochastic Matrix

1. A vector is a **probability vector** (also called a **stochastic vector**) if none of the values in the vector is negative, and the sum of the values in the vector is 1.

2. A matrix is a **stochastic matrix** if none of the values in the matrix is negative, and the sum of the values of every column in the matrix is 1.

30. Transition Matrix of a Directed Graph

Let G be a directed graph.

1. Suppose that G has N vertices. Let $v_1, \ldots, v_N$ be the vertices of G, and let $k_1, \ldots, k_N$ denote the degrees of the vertices, respectively. The **transition matrix** of G is the $N \times N$ matrix that has the (j, i) entry in the matrix equal to $\frac{1}{k_i}$ if there is an edge from v_i to v_j and equal to 0 if there is no edge from v_i to v_j.

2. The transition matrix of G is a stochastic matrix.

31. PageRank Algorithm Using Matrices

Let G be a directed graph. This algorithm, called the **PageRank Algorithm**, ranks the vertices of G by assigning a number to each vertex, where the vertices are then ranked from highest assigned number to lowest assigned number.

In the matrix version of the PageRank algorithm, rather than directly assigning a number to each vertex, a probability vector is found that has as many entries as there are vertices in G; the first entry of the probability vector is then assigned to the first vertex, and so on.

The matrix version of the PageRank algorithm works by defining an initial probability vector, and then repeatedly multiplying it by a certain stochastic matrix, until the desired probability vector is found.

Let p be a number between 0 and 1. The number p is typically 0.85.

1. Suppose that G has n vertices. The initial probability vector v is the vector with n entries, and with each entry equalling $\frac{1}{n}$.

2. Let A be the transition matrix of the graph G. The matrix A is an $n \times n$ stochastic matrix.

3. Let B be the $n \times n$ matrix that has all its entries equal to $\frac{1}{n}$. The matrix B is an $n \times n$ stochastic matrix.

4. Let $M = pA + (1 - p)B$. The matrix M is an $n \times n$ stochastic matrix.

5. Compute Mv, then compute M^2v, then compute M^3v, and so on. Keep going, until it appears that the resulting vector does not change with each new multiplication by M. All these vectors are probability vectors.

6. The probability vector that does not change when multiplied by M, found in the previous step, is used to assign a number to each vertex, where the first entry in the probability vector is assigned to the first vertex, and so on.

7. A quicker way to find the desired probability vector is to find a probability vector $\bar{v}$ such that $M\bar{v} = \bar{v}$, though doing so requires a bit more algebra than the above method.