Northeastern University
College of Engineering

INFO 6205 Program Structure and Algorithms

Course Information

Course Title: Program Structure and Algorithms
Course Number: INFO 6205
Term and Year: Spring 2022
Credit Hour: 4
Course Format: On-Ground

Instructor Information

Full Name: Robin Hillyard
Email Address: r.hillyard@northeastern.edu

Course Prerequisites

Graduate level INFO 5100 Minimum Grade of B- or Graduate level CSYE 6200 Minimum Grade of B-

Course Description

Presents data structures and related algorithms, beginning with a brief review of dynamic memory allocation. Discusses the fundamental data structures in detail, including the abstract representation, supporting algorithms, and implementation methods. Focuses on understanding the application of the abstract data structure and the circumstances that affect implementation decisions. Covers lists, stacks, queues, trees, hash tables, and graphs. Covers recursion and searching and sorting algorithms in detail. Emphasizes data abstraction and encapsulation in code design. Explores external storage structures, time permitting.

Standard Learning Outcomes

Learning outcomes common to all College of Engineering Graduate programs:

1. *An ability to identify, formulate, and solve complex engineering problems.*
2. *An ability to explain and apply engineering design principles, as appropriate to the program's educational objectives.*
3. *An ability to produce solutions that meet specified end-user needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors.*

The Information Systems Program accepts students of different engineering backgrounds with minimum programming skills and produces first class Information Systems engineers that operate at the intersection of real-world complexity, software development, and IT management. Graduating students will be able to construct end-to-end advanced software applications that meet business needs.

Specific Learning Outcomes for the Information Systems program:

1. *Create a strong technical foundation through diverse, high-level courses*
2. *Built crucial interpersonal skills needed to succeed in any industry*

3. *Foster a deep level of applied learning through project based case studies*

Course Outcomes:

1. Develop an approach to problem-solving.
2. Utilize different strategies to develop and improve various algorithms.
3. Articulate the relationship between data structures, algorithms, and invariants.
4. Choose the most effective utility (e.g., quicksort) for a particular programming task.
5. Prepare for a coding interview.

Required Materials:

- *Algorithms*, 4th Edition by Robert Sedgewick and Kevin Wayne, Addison-Wesley Professional, 2011, ISBN 9780321573513
- *Java Integrated Development* Environment: here are installation instructions
- (Optional) *Algorithms*, Dasgupta, Papadimitriou and Vazirani, McGraw-Hill Higher Education, 2008, ISBN 9780073523408

What is this class all about?

- Mostly it's about solving problems by computer.
- You will learn about data structures, the rules that govern them (invariants), and the algorithms that operate on them.
- We will take a practical approach, that's to say we will treat the subject matter with an "engineering" mindset.
- We will get a little mathematical where it's appropriate because the foundation of everything we cover is, essentially, mathematics.

Search Problems

- Most of the problems we set out to solve will be "search problems."
- In computer science terminology, a search problem is one where we will look for a solution within a potentially vast search space until we find one which satisfies our criteria.
- An example would be sorting: the solution we seek is one where all of the elements are in some predefined order.

Of course, this is just an idealized way of classifying these problems—we are usually a lot smarter than that.

Practicalities

The implementation language of this online version of the class is Java (8 or later). The subject material should ideally be independent of any particular language. But for practical reasons, we use Java. Examples and assignments are found in the github repository: https://github.com/rchillyard/INFO6205 (Java classes only).

Module 1: Solving Problems: The Use of Data Structures, Algorithms, and Invariants

Module Overview
In this module, we will explore the nature of the kinds of problems we can solve using computers. We will learn about different strategies for solving these problems: identifying invariants, using brute force, dividing or simplifying the problem into easier problems, and taking advantage of any existing order.

Learning Objectives
By the end of this module, you will be able to do the following: • Develop techniques for solving problems. • Detect invariants when solving problems. • Apply the concept of reduction at an elementary level. • Prove a recurrence relation by mathematical induction. • Sort a small array using insertion sort. • Identify a "search problem," and show how a dictionary saves time.

Reading and Resources	
Required Resources	**Description**
Sedgewick/Wayne, Algorithms 4/e, 6. Context, section "Reduction", pages 903-9	This section explores the concept of reduction in more detail.
IntelliJ IDEA	Install the latest community edition.
https://github.com/rchillyard/INFO6205	Clone the repository and run edu.neu.coe.info6205.sort.simple.InsertionSortTest. Study the insertion sort method being tested.
Optional Resources	**Description**
https://youtu.be/ROalU379l3U	Insert-sort with Romanian folk dance
Dasgupta, Papadimitriou, Vazirani, Algorithms; McGraw-Hill, 2008, ISBN: 978-0-07-352340-8	Chapter 1. This book is really an excellent book about algorithms. The style is very different from Sedgewick and Wayne. And it's a relatively light-weight paperback.
https://youtu.be/QvuQH4_05LI	ThreeBlueOneBrown on Problem-solving

Module Content and Tasks

Title: 1.1 Solving Puzzles
Lesson Description: In this lesson, we will take a look at strategies for solving problems and puzzles by computer - with special emphasis on the concept of invariants.

Title: 1.2 Recursion and Reduction
Lesson Description: In this lesson we take an early look at the concept of reduction, which is fundamental to the efficient solution of most computer problems. And, we examine one of the most common strategies for implementing such solutions: recursion.

Title: 1.3 Search Problems
Lesson Description: We learn some definitions in this lesson, including that of a search problem.

Title: 1.4 Sorting
Type: Embedded Slideshow or link to PowerPoint file
Lesson Description: This lesson introduces sorting using insertion sort as an exemplar.

Module 2: Complexity

Module Overview
In order to understand the behavior and thus the cost of running algorithms, it is important to be able to place bounds on the efficacy of a solution or the difficulty of a problem. These are the roles of the Big O and Big Omega functions respectively. So, we will develop polynomial functions of N, the size of a problem. Because these functions usually involve logarithms, we will start with an in-depth look at logarithms. Next, we must recognize that the time to solve a problem may have as much to do with its memory requirements as the number of instructions executed. But, we must recognize that predicting the behavior from a theoretical viewpoint is likely to be at best an approximation. To really understand, we must benchmark our algorithms, and prior to that, test them. Finally, coming full-circle to the minimum complexity of the problem itself, we need some way to estimate that complexity--and for that we will learn about information entropy.
Learning Objectives
By the end of this module, you will be able to do the following: • Plan how to plant 10 trees & design a 10-stage rocket. • Use an understanding of memory and processor cycles to improve running time. • Time and test the performance of an algorithm.
• Classify problems according to their "order of growth." • Estimate the entropy of a search problem.

Reading and Resources	
Required Resources	**Description**
Sedgewick/Wayne, Algorithms 4/e, 1 Fundamentals, 1.4 Analysis of Algorithms	Analysis of Algorithms. This goes into much more detail on this topic.
Optional Resources	**Description**
Dasgupta, Papadimitriou, Vazirani, Algorithms; McGraw-Hill, 2008, ISBN: 978-0-07-352340-8	Section 0.3 Big-O notation
https://youtu.be/v4cd1O4zkGw	Hacker Rank on Big-O notation (some cute graphics) 8:30
https://youtu.be/4PDoT7jtxmw	2Blue1Brown on Natural Logarithms. Good but long. Definitely optional.

Module Content and Tasks

Title: 2.1 Logarithms
Lesson Description: A detailed look at the logarithm function--the importance of which cannot be overemphasized.

Title: 2.2 Space vs Time
Lesson Description: Where does all the time go? This lesson looks at both our natural enemies: large numbers of instruction cycles and large numbers of memory accesses. Which is worse? The answer may surprise you.

Title: 2.3 Benchmarking and Test-Driven Development
Lesson Description: For any serious study of the behavior of an algorithm, it's important to understand as much as possible about the mathematical basis of the solution. However, it's even more important to back this up with benchmarking--and of course testing. As we'll see in this lesson, no change should ever be made to optimize software without both testing and benchmarking the result.

Title: 2.4 Asymptotic Notation
Lesson Description: In this lesson, we will look at--and try to understand--the important concepts of Big-O, Big-Omega, and their relatives.

Title: 2.5 Entropy
Lesson Description: No study of problem solving and its mathematical analysis can be complete without an understanding of the root cause of the complexity: entropy. Understanding entropy can help us determine the theoretically best possible solution.

Module 3: Abstract Data Types, Arrays, Objects, and Primitives

Module Overview
In this module, we introduce the concept of an abstract data type--that's to say a pluggable component which can be employed by an application. We introduce stacks (which you are probably already familiar with), queues, bags, etc.

Learning Objectives

- Explain why an array is an important data structure.
- Initialize and maintain an array of objects or primitives.
- Design a simple data structure.
- Define an API.
- Design and use a linked list/stack.
- Design and use a bag.
- Design and use both a regular queue and a priority queue.
- Differentiate between an object reference and a primitive.

Reading and Resources	
Required Resources	**Description**
Sedgewick/Wayne, Algorithms 4/e, 1. Fundamentals: 1.2 Data Abstraction.	This chapter explores data abstraction in general.

Sedgewick/Wayne, Algorithms 4/e, 1. Fundamentals: 1.3 Bags, Stacks and Queues.	This chapter explores the abstract data types: bags, stacks and queues.

Module Content and Tasks

Title: 3.1 Abstract Data Types
Lesson Description: We will learn about the importance of the abstract data type as a component of applications.

Title: 3.2 LinkedList/Stack
Lesson Description: The first ADT we study is perhaps the most familiar: the linked list. We will learn how the stack is essentially the same thing but which presents a different application programming interface.

Title: 3.3 Bag
Lesson Description: This lesson introduces the lesser-known but extremely important ADT called the Bag.

Title: 3.4 Queue
Lesson Description: The double-ended queue (or deque) is also a very important ADT. Like its counterpart in normal life, it plays an essential role in matching supply with demand and ensuring that clients are served on a first-come first-served basis.

Title: 3.5 Priority Queue
Lesson Description: A variation on the double-ended queue, the priority queue has surprisingly many applications, especially in the world of Big Data. Instead of serving on a first-come first-served basis, it serves the most important (highest priority) first.

Title: 3.6 Objects vs Primitives in Java
Type: Embedded Slideshow or link to PowerPoint file
Lesson Description: All languages have their idiosyncrasies and Java is no exception. But, if you strive to write very efficient code, you must understand distinctive aspects of the language covered in this lesson.

Module 4: Sorting, Shuffling & Selection

Module Overview

The "Dictionary principle" allows us to speed up searching, providing that we have an efficient method of sorting. Sorting comprises a large proportion of this course because it is so important. We cover all of the most important sorting algorithms here, leaving a few of the more advanced ones to Module 8. Additionally, we look at shuffling, or un-sorting, and selection, which essentially is finding the kth element of a collection, without having to sort it first.

Learning Objectives

By the end of this module, you will be able to do the following:

- Code a typical comparison method for a class of objects and describe the ways by which objects may be compared.
- Enumerate the rules of total order.

- Code selection sort and describe its complexity.
- Code shell sort and describe its complexity.
- Code merge sort and describe its complexity.
- Code quicksort and describe its complexity.
- Code heap sort and describe its complexity.
- Describe the complexity of QuickSelect.
- Code the Knuth shuffle and describe its complexity.

Reading and Resources	
Required Resources	**Description**
Sedgewick/Wayne, Algorithms 4/e, 2 Sorting, 2.1 Elementary Sort	This section provide details description and analysis of elementary sort.
Sedgewick/Wayne, Algorithms 4/e, 2 Sorting, 2.2 Merge sort	This section focuses on Merge sort and various approaches of merge sort.
Sedgewick/Wayne, Algorithms 4/e, 2 Sorting, 2.3 Quicksort	This section explains multiple partitioning techniques and quicksort.
Optional Resources	**Description**
https://www.toptal.com/developers/sorting-algorithms	This page shows visualization of multiple sorting algorithms.
http://me.dt.in.th/page/Quicksort/#disqus_thread	In-depth discussion of quicksort.
https://youtu.be/lyZQPjUT5B4	**Bubble Sort by folk dance.**
https://youtu.be/CmPA7zE8mx0	**Shell Sort by folk dance using gaps of 5, 3, 1.**
https://youtu.be/XaqR3G_NVoo	**Merge Sort by folk dance.**
https://youtu.be/ywWBy6J5gz8	**Quicksort by folk dance.**

Module Content and Tasks

Title: 4.1 Comparison Sorts
Lesson Description: General introduction to comparison sorts, invariants, and swaps.

Title: 4.2 Selection Sort
Lesson Description: Description of selection sort

Title: 4.3 Shell Sort
Lesson Description: Description of shell sort: how to reduce the number of inversions before sorting

Title: 4.4 Merge Sort
Lesson Description: Divide and conquer-based sorting: Merge sort

Title: 4.5 Quicksort
Lesson Description: Divide and conquer-based sorting: Quicksort

Title: 4.6 Heap Sort
Lesson Description: Binary heaps and heap sort.

Title: 4.7 Quickselect
Lesson Description: Order statistics without getting a complete order.

Title: 4.8 Shuffling
Lesson Description: Shuffling – the opposite of sorting.

Module Summary Chart
Title: Module 4 Interactive Flowchart for Choosing a Sort Algorithm **URL:** https://drive.google.com/file/d/1e5DGj9xhwppriiCNqAxPs9QWQmbK-AlQ/view?usp=sharing

Module 5: Searching

Module Overview
Information retrieval is one of the most important tasks for computers. How can we store information in such a way that retrieval of the information is fast and uses as little memory as is reasonably possible? How do we deal with objects that we can compare and objects that we cannot compare? How do we retrieve information when getting a block of data is extremely slow (as with a hard disk, or a network lookup, for instance)? These questions are addressed in this important module.

Learning Objectives
By the end of this module, you will be able to do the following: • Code binary search and describe its complexity. • Analyze the performance of a binary search tree. • Differentiate the 2-3 tree from the left-leaning red-black tree. • Identify situations for which a B-tree search is applicable, and state the advantages of that search. • Discuss the required features of a hash function for searching. • Code and analyze separate chaining. • Code and analyze open addressing.

Reading and Resources

Required Resources	**Description**
Sedgewick/Wayne, Algorithms 4/e, sections 3.1, 3.2, 3.3, 3.4 (pages 362-485)	A key value pair data structure introduction
Sedgewick/Wayne, Algorithms 4/e, sections 3.2	Detailed description of Binary search tree and its applications
Sedgewick/Wayne, Algorithms 4/e, sections 3.3, 3.4	Balanced Search Tree, Hash Table

Optional Resources	Description
Sedgewick/Wayne, Algorithms 4/e, sections 3.5 (pages 486-513)	Applications of Searching
Sedgewick/Wayne, Algorithms 4/e, pages 866-874	B-trees

Module Content and Tasks

Title: 5.1 Introduction to Searching
Lesson Description: Searching basics: lists, arrays.

Title: 5.2 Binary Search Trees
Lesson Description: How an extra degree of freedom can allow us to efficiently store and retrieve keys.

Title: 5.3 Balanced Search Trees
Lesson Description: Adding another degree of freedom to allow us to balance a search tree.

Title: 5.4 B-Trees
Lesson Description: Extending the concept of balanced trees.

Title: 5.5 Hash Functions
Lesson Description: How to get the benefits of an array with generalized keys.

Title: 5.6 Separate Chaining
Lesson Description: Chaining: the simplest implementation of a hash table.

Title: 5.7 Open Addressing
Lesson Description: Also known as linear probing, a slightly more efficient hash table if it's well managed.

Module Summary Chart

Title: Module 5 Interactive Flowchart for Choosing an ADT for Search
Type: Interactive Learning Object
URL: https://drive.google.com/file/d/1Pu2XiHH_rjKu1iMzi19jQovBEiFw_u1_/view?usp=sharing

Module 6: Compression

Module Overview

Space (memory) can be as important as the number of instructions when it comes to the performance of algorithms. Compressing data for transmission over a network, for instance, can make a significant difference in the time taken for messages to be passed. We will look in detail only at Huffman coding, but the textbook also covers run-length encoding.

Learning Objectives
By the end of this module, you will be able to do the following: • Describe the role of compression algorithms in data storage, communication, and processing. • Encode and decode by Huffman coding.

Reading and Resources	
Required Resources	**Description**
Sedgewick/Wayne, Algorithms 4/e, 5.5 Data Compression, pages 826-839	These pages explore Huffman Coding and Tries.
Optional Resources	**Description**
Sedgewick/Wayne, Algorithms 4/e, 5.5 Data Compression, pages 810-850	This section additionally explores Run-length Encoding and LZW Compression.

Module Content and Tasks

Title: 6.1 Huffman Codes Lesson Description: Message compression by Huffman coding.

Module 7: Graphs and Trees

Module Overview
This module covers those aspects of graph theory suitable for a general course such as INFO6205. Graphs are increasingly important data structures for the modern world and, because graphs tend to be very large in terms of the numbers of edges, it is extremely important to design efficient algorithms.

Learning Objectives
By the end of this module, you will be able to do the following: • Describe the importance and applications of graphs and trees. • Differentiate between a tree and a graph which is not a tree. • Code and analyze depth-first and breadth-first traversal of trees and graphs. • Differentiate between undirected and directed graphs. • Design data structures for undirected and directed graphs. • Explain how the connected components algorithm is used to compute connected components for a given graph. • Code and analyze Kosaraju's algorithm. • Code and analyze Dijkstra's algorithm.

Reading and Resources	
Required Resources	**Description**
Sedgewick/Wayne, Algorithms 4/e, Chapter 4, Graphs: pages 514-693	These chapters explore everything about graphs.
https://www.maa.org/press/periodicals/convergence/leonard-eulers-solution-to-the-konigsberg-bridge-problem	The Seven Bridges of Königsberg

Optional Resources	Description
http://www1.coe.neu.edu/~rhillyard/Tunnel%20article.pdf	Application of Kruskal's algorithm--part of lesson 7.5
https://tinkerpop.apache.org	Apache Tinker Pop Graph Computing Framework

Module Content and Tasks

Title: 7.1 Introduction to Trees and Graphs
Lesson Description: General description of trees and graphs.

Title: 7.2 Depth-first vs Breadth-first
Lesson Description: The two primary strategies for traversing graphs and trees.

Title: 7.3 Undirected and Directed Graphs
Lesson Description: Graph properties and algorithms.

Title: 7.4 Connected and Strong Components
Lesson Description: Connectivity in graphs – undirected and directed.

Title: 7.5 Minimum Spanning Trees
Lesson Description: How to define a minimum spanning tree.

Title: Kruskal's Algorithm, NEU Tunnel System
Type: Explainer Video

Title: 7.6 Shortest Paths
Lesson Description: Calculating the shortest path(s) from A to B using Dijkstra's algorithm.

Module 8: Advanced Sorts

Module Overview

This module covers advanced sorts--mostly sorts that either don't use comparison at all or, if they do, use it in some hybrid algorithm. Module 8 also introduces the Group Project.

Learning Objectives

By the end of this module, you will be able to do the following:

- Code and analyze pre-sorting algorithms.
- Code and analyze non-comparison-based sorting.
- Demonstrate teamwork and cooperation skills.

Task List
• Lesson 8.1 Intro Video, lesson content, and Check Your Knowledge • Lesson 8.2 Intro Video, lesson content, and Check Your Knowledge • Coding Assignment • Summary Video • Start Team Project

Module Content and Tasks

Title: 8.1 Bucket Sort and Husky Sort Lesson Description: Hybrid sorts
Title: 8.2 Radix Sorts Lesson Description: non-comparison sorts.

Module 9: Finite Automata & Classifying Search Problems

Module Overview
This module covers some advanced theoretical topics that don't fit neatly into any of the other modules.
Learning Objective
By the end of this module, you will be able to do the following: • Code and use regular expressions. • Understand the relationship between regular expressions and finite automata. • Distinguish between P, NP, etc. • Demonstrate teamwork and cooperation skills.

Reading and Resources	
Required Resources	**Description**
Sedgewick/Wayne, Algorithms 4/e, 5.4 Regular Expressions, pages 788-809	These pages explore Regular Expressions, as well as deterministic and non-deterministic finite automata..
Sedgewick/Wayne, Algorithms 4/e, Reduction, pages 903-921	Reduction, Intractability

Module Content and Tasks

Lesson 9.1
Title: Regular Expressions Lesson Description: Finite State Machines and their application to parsing.

Lesson 9.2
Title: DFAs and NFAs Lesson Description: Deterministic and non-deterministic finite automata

Lesson 9.3
Title: Classifying Search Problems Lesson Description: P and NP

Project and Optional Module 10: Advanced Problem Solving

Module Overview
This module covers specialized topics that are not treated in standard textbooks but which you should have at least some knowledge about.

Learning Objectives
By the end of this module, you will be able to do the following: • Design a non-deterministic algorithm for solving an NP-complete problem. • Differentiate between greedy algorithms and dynamic programming. • Demonstrate teamwork and cooperation skills.

Reading and Resources	
Required Resources	**Description**
Sedgewick/Wayne, Algorithms 4/e, pages tk	These chapters explore .

Module Content and Tasks

Lesson 10.1
Title: Non-deterministic Programming Lesson Description: Introduction to Genetic Algorithms

Lesson 10.2
Title: Dynamic Programming Lesson Description: A quick look at dynamic programming

Module 11: Guide to Interviews

Module Overview
This module does not directly relate to anything in the textbook, nor will it appear in the final exam. Interviews for software development positions have become harder and more competitive in recent decades. Gone are the days when the interviewer would ask just one or two softball questions about languages you claimed to know. In short, you are expected to apply everything you learned in this course during an intense verbal examination that might last an hour. There is no substitute for practice. The more you solve internet problems, the more comfortable you will be under pressure. This module gives you a few pointers to help you when the moment comes.

Learning Objectives
By the end of this module, you will be able to do the following: • Conduct an appropriate coding interview. • Debug your own programs. • Apply problem-solving skills to a coding case study. • Demonstrate teamwork and cooperation skills.

Reading and Resources	
Required Resources	**Description**
None	
Optional Resources	**Description**
https://www.amazon.com/dp/098478 2850/ref=cm_sw_em_r_mt_dp_U_2f dhFbEWAE53S	Cracking the Coding Interview by Gayle Laakman McDowell
https://medium.com/@AndyyHope/softw are-engineering-interviews-744380f4f2af	Excellent article on preparing for interviews at companies like Facebook.
https://en.wikipedia.org/wiki/The_Hitchhi ker's_Guide_to_the_Galaxy	**The increasingly inaccurately described Hitchhikers' trilogy.** **The cover of the book is DON'T PANIC**

Module Content and Tasks

Lesson 11.1
Title: DON'T PANIC Lesson Description: The Hitchhiker's Guide to Interviews 11.1

Lesson 11.2
Title: Interview Case Studies Lesson Description: A look at one or two example problems

Module 12: Final Exam & Submit Team Project

End-of-Course Evaluation Surveys

Your feedback regarding your educational experience in this class is very important to the College of Professional Studies. Your comments will make a difference in the future planning and presentation of our curriculum.

At the end of this course, please take the time to complete the evaluation survey at **https://neu.evaluationkit.com**. Your survey responses are **completely anonymous and confidential**. For courses 6 weeks in length or shorter, surveys will be open one week prior to the end of the courses; for courses greater than 6 weeks in length, surveys will be open for two weeks. An email will be sent to your HuskyMail account notifying you when surveys are available.

Academic Integrity

A commitment to the principles of academic integrity is essential to the mission of Northeastern University. The promotion of independent and original scholarship ensures that students derive the most from their educational experience and their pursuit of knowledge. Academic dishonesty violates the most fundamental values of an intellectual community and undermines the achievements of the entire University.

As members of the academic community, students must become familiar with their rights and responsibilities. In each course, they are responsible for knowing the requirements and restrictions regarding research and writing, examinations of whatever kind, collaborative work, the use of study aids, the appropriateness of assistance, and other issues. Students are responsible for learning the conventions of documentation and acknowledgment of sources in their fields. Northeastern University expects students to complete all examinations, tests, papers, creative projects, and assignments of any kind according to the highest ethical standards, as set forth either explicitly or implicitly in this Code or by the direction of instructors.

Go to http://www.northeastern.edu/osccr/academic-integrity-policy/ to access the full academic integrity policy.

Student Accommodations

Northeastern University and the Disability Resource Center (DRC) are committed to providing disability services that enable students who qualify under Section 504 of the Rehabilitation Act and the Americans with Disabilities Act Amendments Act (ADAAA) to participate fully in the activities of the university. To receive accommodations through the DRC, students must provide appropriate documentation that demonstrates a current substantially limiting disability.

For more information, visit http://www.northeastern.edu/drc/getting-started-with-the-drc/.

Library Services

The Northeastern University Library is at the hub of campus intellectual life. Resources include over 900,000 print volumes, 206,500 e-books, and 70,225 electronic journals.

For more information and for Education specific resources, visit http://subjectguides.lib.neu.edu/edresearch.

24/7 Blackboard Technical Help

For immediate technical support for Blackboard, call 617-373-4357 or emailhelp@northeastern.edu

Within Blackboard, open a support case via the red support button on the right side of the screen, click Create Case

myNortheastern, e-mail, and basic technical support
Visit the Information Technology Services (ITS) Support Portal
Email: help@northeastern.edu
ITS Customer Service Desk: 617-373-4357

Diversity and Inclusion

Northeastern University is committed to equal opportunity, affirmative action, diversity and social justice while building a climate of inclusion on and beyond campus. In the classroom, member of the University community work to cultivate an inclusive environment that denounces discrimination through innovation, collaboration and an awareness of global perspectives on social justice.

Please visit http://www.northeastern.edu/oidi/ for complete information on Diversity and Inclusion

TITLE IX

Title IX of the Education Amendments of 1972 protects individuals from sex or gender-based discrimination, including discrimination based on gender-identity, in educational programs and activities that receive federal financial assistance.

Northeastern's Title IX Policy prohibits Prohibited Offenses, which are defined as sexual harassment, sexual assault, relationship or domestic violence, and stalking. The Title IX Policy applies to the entire community, including male, female, transgender students, faculty and staff.

In case of an emergency, please call 911.

Please visit www.northeastern.edu/titleix for a complete list of reporting options and resources both on- and off-campus.